UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/727,190	12/03/2003	Heonchul Park		5182

7590 03/08/2006

Jay Chesavage
3833 Middlefield Rd.
Palo Alto, CA 94303

EXAMINER	
GEIB, BENJAMIN P	
ART UNIT	**PAPER NUMBER**
2181	

DATE MAILED: 03/08/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	10/727,190	PARK, HEONCHUL
	Examiner	Art Unit
	Benjamin P. Geib	2181

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 December 2003 and 15 October 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-40* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-40* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *03 December 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *10/15/2004*.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-40 have been examined.

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Application on 12/03/2003 and Information Disclosure Statement on 10/15/2004.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claims 8, 14, and 15 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

5. Referring to claim 8, the recited limitation "pipeline core comprises a subset of said stages on one thread and remaining said stages on said other thread" is indefinite because it is unclear how a "stage" can be "on" a thread.

6. Referring to claim 14, the recited limitation "decode a device" is indefinite because it is unclear how a physical apparatus, a device, can be decoded.

7. Referring to claim 15, the recited limitation "devices are decoded" is indefinite for the reason stated regarding claim 14.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1-40 are rejected under 35 U.S.C. 103(a) as being unpatentable over the admitted prior art (Fig. 1 and pages 5-6) of application 10/727190 (Herein referred to as APA) in view of Hokenek et al., U.S. Patent No. 6,842,848 (Herein referred to as Hokenek).

10. Referring to claim 1, APA has taught a processor comprising:

a register set *[APA; Fig. 1, component 24]* comprising a plurality of registers *[APA; page 6, lines 7-11]*;

an n-way register set controller *[APA; Fig. 1, component 26]* coupled to said register set *[APA; page 6, lines 22-23]*;

a Fetch Address Stage *[APA; Fig. 1, component 14]* for the generation of Program Memory Addresses *[APA; page 5, line 23 – page 6, line 1]*;

a Program Access Stage *[APA; Fig. 1, component 22]* for receiving Program Memory Data associated with said Program Memory Addresses *[APA; page 6, lines 1-3]*;

a Decode Stage *[APA; Fig. 1, component 28]* for converting said Program Memory Data into instructions *[APA; page 6, lines 3-7]*, said Decode Stage coupled to said n-way register set controller *[APA; See Fig. 1]*;

a First Execution Stage *[APA; Ex 1; Fig. 1, component 34]* for handling a multiply class of instruction received from said Decode Stage *[APA; page 6, lines 11-13]*;

a Second Execution Stage *[APA; Ex 2; Fig. 1, component 36]* for handling an Arithmetic Logical Unit class of instructions received from said Decode Stage *[APA; page 6, lines 11-16]*, said Second Execution Stage also coupled to said n-way register set controller *[APA; See Fig. 1]*;

a Memory Access Stage *[APA; Fig. 1, component 38]* for handling reading and writing of external memory *[APA; page 6, lines 17-22]*;

a Write Back Stage *[APA; Fig. 1, component 44]* coupled to said register set controller for writing data to said register set *[APA; page 6, lines 22-23]*;

each said Stage performing an operation during a Stage Cycle *[APA; page 5, lines 5-6]*;

APA has not explicitly taught that the processor is a multithread processor for processing a plurality of threads comprising:

a thread ID generator producing a unique thread indication for each said thread;

a plurality of register sets, one said register set for each said thread;

wherein the register set controller is coupled to said plurality of register sets and simultaneously handles multiple read or write requests for one or more of said unique threads; and

wherein said Thread ID value alternating from one stage cycle to the next.

Hokenek discloses a multithread processor for processing a plurality of threads *[Hokenek; column 4, lines 26-34]* comprising:

a thread ID generator producing a unique thread indication for each thread *[Hokenek; column 7, lines 39-52]*

a plurality of register sets *[Hokenek 2 (see below); Fig. 7, components T_0-T_3]*, one said register set for each said thread *[Hokenek 2 (see below); column 7, lines 3-26]*

[Register file access is described in Hokenek et al., U.S. Patent No. 6,904,511 (Application Ser. No. 10/269,373; Herein referred to as Hokenek 2), which has been incorporated by reference into Hokenek (See column 1, lines 6-14)];

wherein a register set controller *[Hokenek 2; selection circuitry; Fig. 7, components 704]* is coupled to said plurality of register sets and simultaneously handles multiple read or write requests for one or more of said unique threads *[Hokenek 2; column 4, lines 43-54]*; and

wherein said Thread ID value alternating from one stage cycle to the next *[Hokenek; The Thread ID value alternates among the N Thread IDs from one stage cycle to the next; See Fig. 5]*

At the time the invention was made, it would have been obvious to a person of ordinary skill in the art to modify the processor of APA to support multithreading as taught by Hokenek, comprising a thread ID generator producing a unique thread

indication for each said thread; a plurality of register sets, one said register set for each said thread; wherein the register set controller is coupled to said plurality of register sets and simultaneously handles multiple read or write requests for one or more of said unique threads; and wherein said Thread ID value alternating from one stage cycle to the next.

The suggestion/motivation for doing so would have been that doing so advantageously provides "enhanced processor concurrency and reduced likelihood of thread stalling" *[Hokenek; column 4, lines 26-34]*.

Therefore, it would have been obvious to combine Hokenek with APA to obtain the invention as specified in claim 1

11. Referring to claim 2, APA and Hokenek have taught the processor of claim 1 where a pipeline core is formed by stages in succession: said Fetch Address stage, said Program Access stage, said Decode stage, said First Execution stage, said Second Execution stage, said Memory Access stage, and said Write Back stage *[APA; See Fig. 1]*.

12. Referring to claim 3, APA and Hokenek have taught the processor of claim 1 where said n-way register set controller simultaneously receives at least one of read requests from said Decode stage, read and write requests from said Second Execution stage, or write requests from said Write Back stage *[APA; See Fig. 1]*.

13. Referring to claim 4, APA and Hokenek have taught the processor of claim 1 where said Memory Access stage is coupled to a memory controller *[APA; Fig. 1, component 42]*.

14. Referring to claim 5, APA and Hokenek have taught the processor of claim 4 where said memory controller issues a stall signal *[APA; Fig. 1, component 46]* when receiving a memory request to an external memory *[APA; page 5, lines 11-22]*.

15. Referring to claim 6, APA and Hokenek have taught the processor of claim 4 where said memory controller issues a stall signal *[APA; Fig. 1, component 46]* when receiving a memory read request to an external memory *[APA; page 5, lines 11-22]*.

16. Referring to claim 7, APA and Hokenek have taught the processor of claim 4 where said memory controller issues a stall signal which lasts an interval from receiving a memory read request to receiving requested data form said external memory *[APA; page 5, lines 11-22]*.

17. Referring to claim 8, APA and Hokenek have taught the processor of claim 1 where said pipeline core comprises a subset of said stages on one thread and remaining said stages on said other thread *[The threads issue an instruction/cycle in round-robin fashion (Hokenek; See Fig. 3; column 4, lines 36-62). Therefore, when the number of threads N is 2, a subset of stages will be executing a first thread while the remaining stage are executing a second thread]*.

Referring to claim 9, APA and Hokenek have taught the processor of claim 1 where said first execution stage *[APA; Ex 1]* performs multiply operations and said second execution stage *[APA; Ex 2]* performs non-multiply instructions *[APA; page 6, lines 11-16]*.

18. Referring to claim 10, APA and Hokenek have taught the processor of claim 1 where said decode stage forwards non-multiply operands to said second execution stage *[APA; page 6, lines 11-16; See Fig. 1]*.

19. Referring to claim 11, APA and Hokenek have taught the processor of claim 1 including a program memory *[Hokenek; main memory; column 3, lines 30-34; See Fig. 1]*.

APA and Hokenek have not expressly disclosed that the program memory contains a single instance of a program.

However, Examiner takes Official Notice that having program memory contain a single instance of a program is a conventional and well known means of reducing the amount of memory needed to store a program.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify APA and Hokenek to have program memory contain a single instance of a program since Examiner takes Official Notice of having program memory contain a single instance of a program as a conventional and well known means of reducing the amount of memory needed to store a program.

20. Referring to claim 12, APA and Hokenek have taught the processor of claim 1 where said thread ID can be read by each said thread *[Hokenek; The thread ID, as stored in the global NTID register, can be read by each thread; column 7, lines 14-24]*.

21. Referring to claim 13, APA and Hokenek have taught the processor of claim 1 where each said thread reads said thread ID to perform thread operations which are independent *[Each thread, or context, reads the thread ID and performs operations that*

are specific to that context and, therefore, independent of other threads; Hokenek; column 6, line 55 – column 7, line 24].

22. Referring to claim 14, APA and Hokenek have taught the processor of claim 1 where thread ID is used along with an address to decode a device in a memory map *[The thread ID is used along with an address to access a memory map; Hokenek 4 (see below); column 6, line 51 – column 7, line 11; Fig. 6].*

[Details concerning memory access using a thread ID are included in Hokenek et al., U.S. Patent No. 6,925,643 (Application Ser. No. 10/269,247; Herein referred to as Hokenek 4), which has been incorporated by reference into Hokenek (See column 1, lines 6-14)]

23. Referring to claim 15, APA and Hokenek have taught the processor of claim 1 where devices are decoded in a memory map based on address only *[Hokenek 4; See Background].*

24. Referring to claim 16, APA and Hokenek have taught the processor of claim 1 where said Decode stage performs decoding of instructions for said multiply class of instruction and said arithmetic logical unit class of instructions *[APA; page 6, lines 3-16].*

APA and Hokenek have not expressly disclosed that decoding is done incrementally with multiply class instructions being decoded in a first stage and arithmetic logical unit class instructions decoded in a second stage.

However, Examiner takes Official Notice that incrementally decoding instructions with multiply class instructions being decoded in a first stage and arithmetic logical unit

class instructions decoded in a second stage is a conventional and well known means of reducing stage cycle time.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify APA and Hokenek to incrementally decode instructions with multiply class instructions being decoded in a first stage and arithmetic logical unit class instructions decoded in a second stage since Examiner takes Official Notice of incrementally decoding instructions with multiply class instructions being decoded in a first stage and arithmetic logical unit class instructions decoded in a second stage as a conventional and well known means of reducing stage cycle time.

25. Referring to claim 17, APA and Hokenek have taught the processor of claim 16 where if one of said multiply class of instructions requires a register operand, said operand is provided from said registers to said decode stage *[APA; page 6, lines 8-16; See Fig. 1]*.

26. Referring to claim 18, APA and Hokenek have taught the processor of claim 16 where if one of said arithmetic logical unit class of instructions requires a register operand, said operand is provided from said registers to said first execution stage *[APA; page 6, lines 8-16; See Fig. 1]*.

27. Referring to claim 19, APA and Hokenek have taught the processor of claim 1 where at least one said stage includes an operational clock which is at a higher rate than said clock *[APA; page 5, lines 4-7]*.

28. Referring to claim 20, APA has taught a processor comprising

a plurality of stages *[APA; See Fig. 1]* operating on a stage clock for passing information from stage to stage, each stage including inter-stage storage *[APA; page 5, lines 4-11]*;

a first said stage *[APA; Fetch Address Stage; Fig. 1, component 14]* receiving program counter address information from a program counter *[APA; Fig. 1, component 12]* and delivering said address to a program memory *[APA; page 5, line 23 – page 6, line 1]*;

a second stage *[APA; Program Access Stage; Fig. 1, component 22]* for receiving program data from a program memory *[APA; page 6, lines 1-3]*;

a third stage *[APA; Decode Stage; Fig. 1, component 28]* for performing decode of said program data *[APA; page 6, lines 3-7]*;

a fourth stage *[APA; Ex 1; Fig. 1, component 34]* for performing multiplication operations or decode operations *[APA; page 6, lines 11-13]*;

a fifth stage *[APA; Ex 2; Fig. 1, component 36]* for performing non-multiplication operations *[APA; page 6, lines 11-16]*;

a sixth stage *[APA; Memory Access Stage; Fig. 1, component 38]* for accessing external memory *[APA; page 6, lines 17-22]*;

a seventh stage *[APA; Write Back Stage; Fig. 1, component 44]* for writing results of computations performed in said fourth stage or said fifth stage back to a register set *[APA; page 6, lines 22-23]*;

APA has not explicitly taught that the processor is a multithread processor wherein

inter-stage storage is for thread information associated with a thread ID;

wherein the program counter address information is from a unique program counter for each said thread ID;

wherein the register set being duplicated for each said thread ID; and

wherein each stage receives said thread ID and operates according to a first or second value.

Hokenek discloses a multithread processor for processing a plurality of threads *[Hokenek; column 4, lines 26-34]* wherein

inter-stage storage is for thread information associated with a thread ID *[The inter-stage storage stores data related to a thread of execution (i.e. thread information) (APA; page 5, lines 4-11) and each threads has an associated thread ID (Hokenek; column 7, lines 39-52). Therefore, the inter-stage storage is for thread information associated with a thread ID (Hokenek; column 6, lines 9-13)]*;

wherein the program counter address information is from a unique program counter *[Hokenek 3 (see below); Fig. 1, component 142]* for each said thread ID *[Hokenek 3 (see below); column 3, lines 44-52]*

[Details concerning instruction address generation are included in Hokenek et al., U.S. Patent No. 6,968,445 (Application Ser. No. 10/269,372;

Herein referred to as Hokenek 3), which has been incorporated by reference into Hokenek (See column 1, lines 6-14)];

wherein the register set being duplicated for each said thread ID *[Hokenek 2; Fig. 7, components T_0-T_3; column 7, lines 3-26]*; and

wherein each stage receives said thread ID and operates according to a first or second value *[The NTID register, which is included in all stages, receives the thread ID (Hokenek; column 7, lines 15-24). Therefore, each stage receives the thread ID. When the number of threads N is 2, there are only 2 thread Ids and, therefore, the stage operates according to a first or second value (Hokenek; Fig. 5; column 6, lines 9-13, 51-54)].*

At the time the invention was made, it would have been obvious to a person of ordinary skill in the art to modify the processor of APA to support multithreading as taught by Hokenek, wherein inter-stage storage is for thread information associated with a thread ID; wherein the program counter address information is from a unique program counter for each said thread ID; wherein the register set being duplicated for each said thread ID; and wherein each stage receives said thread ID and operates according to a first or second value.

The suggestion/motivation for doing so would have been that doing so advantageously provides "enhanced processor concurrency and reduced likelihood of thread stalling" *[Hokenek; column 4, lines 26-34].*

Therefore, it would have been obvious to combine Hokenek with APA to obtain the invention as specified in claim 20.

29. Referring to claim 21, APA and Hokenek have taught the multi-threaded processor of claim 20 where said first, third, fifth and seventh stages use one value for said thread ID, and said second, fourth, and sixth stages use a different value for said thread ID *[When the number of threads, N, is 2 two threads alternate issuing instructions. Since the two threads use different IDs, the first, third, fifth, and seventh stages use one thread ID value and the second fourth, and sixth stages use a different value for the Thread ID value; Hokenek; column 6, lines 9-13, 51-54]*.

30. Referring to claim 22, APA and Hokenek have taught the multi-threaded processor of claim 20 where said threads each control execution of a program *[The threads issue instructions (Hokenek; column 6, lines 9-13) and, therefore, control execution of a program]*, and said programs execute independently of each other *[Each thread has its own register set storing the results of program execution (Hokenek 2; Fig. 7, components T_0-T_3; column 7, lines 3-26). Therefore, the programs execute independently of each other]*.

31. Referring to claim 23, APA and Hokenek have taught the multi-threaded processor of claim 22 where one thread may stop execution *[APA; page 5, lines 11-22]*.

APA and Hokenek have not expressly disclosed that the second thread continues execution when the first thread stalls.

However, Examiner takes Official Notice that having a second thread continue execution when a first thread stalls is a conventional and well known means of efficiently using processor execution resources.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify APA and Hokenek to have the second thread continue execution when the first thread stalls since Examiner takes Official Notice of continuing execution of a second thread when a first thread stalls as a conventional and well known means efficiently using processor execution resources.

32. Referring to claim 24, APA and Hokenek have taught the multi-threaded processor of claim 20 where said registers and said stages contain data which is used separately for each said thread ID *[Each thread has a separate register set and, therefore, the stage will contain data used separately for that thread; Hokenek 2; Fig. 7, column 7, lines 3-26]*.

33. Referring to claim 25, APA and Hokenek have taught the multi-threaded processor of claim 20 where said stages alternate between two threads on each said stage clock *[The threads issue an instruction/cycle in round-robin fashion (Hokenek; See Fig. 3; column 4, lines 36-62). Therefore, when the number of threads N is 2, the stages will alternate between two threads on each stage clock]*.

34. Referring to claim 26, APA and Hokenek have taught the multi-threaded processor of claim 20 where said thread-ID identifies a register set *[Hokenek 2; Fig. 7, component Tx; column 7, lines 3-26]* and a program counter *[Hokenek 3; Fig. 1, component 142; column 3, lines 44-52]*.

35. Referring to claim 27, APA and Hokenek have taught the multi-threaded processor of claim 20 where said third stage performs said decode for multiply operations *[APA; page 6, lines 3-16]*.

36. Referring to claim 28, APA and Hokenek have taught the multi-threaded processor of claim 20 where said third stage performs said decode for multiply and non-multiply operations *[APA; page 6, lines 3-16]*.

APA and Hokenek have not expressly disclosed that decoding is done incrementally with multiply class instructions being decoded in the third stage (i.e. a first stage) and arithmetic logical unit class instructions decoded in the fourth stage (i.e. a second stage).

However, Examiner takes Official Notice that incrementally decoding instructions with multiply class instructions being decoded in a first stage and arithmetic logical unit class instructions decoded in a second stage is a conventional and well known means of reducing stage cycle time.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify APA and Hokenek to incrementally decode instructions with multiply class instructions being decoded in the third stage (i.e. a first stage) and arithmetic logical unit class instructions decoded in the fourth stage (i.e. a second stage) since Examiner takes Official Notice of incrementally decoding instructions with multiply class instructions being decoded in a first stage and arithmetic logical unit class instructions decoded in a second stage as a conventional and well known means of reducing stage cycle time.

37. Referring to claim 29, APA and Hokenek have taught the multi-threaded processor of claim 27 where said fourth stage *[APA; Ex 1 Stage]* performs said multiply operations *[APA; page 6, lines 11-16]*.

38. Referring to claim 30, APA and Hokenek have taught the multi-thread processor of claim 28 where said fifth stage *[APA; Ex 2 Stage]* performs said non-multiply operations *[APA; page 6, lines 11-16]*.

39. Referring to claim 31, APA and Hokenek have taught the processor of claim 28 where said non-multiply operations include at least one of rotate, shift, add, subtract, or load *[APA; page 6, lines 11-16]*.

40. Referring to claim 32, APA and Hokenek have taught the multi-thread processor of claim 29 where said multiply operations include multiplication by a constant from one of said registers *[APA; Fig. 1; multiply operations perform multiplication on registers, these values are constants]*.

41. Referring to claim 33, APA and Hokenek have taught the multi-thread processor of claim 30 where said non-multiply operations include addition of a multiply result from said fourth stage *[APA; Fig. 1; the result of a multiply operation is sent to the ALU stage where addition is performed]*.

42. Referring to claim 34, APA and Hokenek have taught the multi-thread processor of claim 20 where said thread ID includes a plurality of values, each said value having at least one register *[Hokenek 2; Fig. 7, component Tx; column 7, lines 3-26]* and a program counter *[Hokenek 3; Fig. 1, component 142; column 3, lines 44-52]*.

43. Referring to claim 35, APA and Hokenek have taught the multi-thread processor claim 20 where said sixth stage *[APA; Memory Access Stage]* said external memory responds in more than one said stage clock cycle *[APA; page 5, lines 11-22; page 6, lines 17-22]*.

44. Referring to claim 36, APA and Hokenek have taught the multi-thread processor of claim 20 where said external memory generates a stall signal for each said thread ID, thereby causing all said stages to store and maintain data for that thread ID until said stall signal is removed by said external memory *[The thread is stalled and completes when external memory removes stall signal. Therefore, it is inherent that the data for that thread ID is stored and maintained; Hokenek; column 5, lines 34-41]*.

45. Referring to claim 37, APA and Hokenek have taught the multi-thread processor of claim 20 where said fifth stage generates an address for a data memory *[APA; See Fig. 1, address signal from Ex2 stage to data memory]*.

46. Referring to claim 38, APA and Hokenek have taught the multi-thread processor of claim 37 where said sixth stage *[APA; Memory Access Stage]* receives and generates data for said data memory *[APA; page 6, lines 17-22]*.

47. Referring to claim 39, APA and Hokenek have taught the multi-thread processor of claim 20 were said thread information storage includes registers which store results from said fifth stage for each said thread ID *[Hokenek 2; Fig. 7, component Tx; column 7, lines 3-26]*.

48. Referring to claim 40, APA and Hokenek have taught the multi-thread processor of claim 20 where said registers *[Hokenek 2; Fig. 7, component Tx]* which store results

from said fifth stage allow a non-stalled thread to continue execution without modifying said stored results *[Hokenek 2; column 7, lines 3-26]*.

Conclusion

49. The following is text cited from 37 CFR 1.111(c): In amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections.

50. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Lauterbach, U.S. Patent Application Publication No. 2003/0046517, has taught a fine-grained multithreaded processor.

Parady, U.S. Patent No. 6,295,600, has taught a coarse-grained multithreaded processor.

Joy et al., U.S. Patent No. 6,938,147, has taught a coarse-grained multithreaded processor.

Katzman et al., U.S. Patent No. 2004/0034759, has taught a fine-grained multithreaded processor.

Loikkanen et al., “A fine-grain multithreading superscalar architecture”, has taught applying fine-grained multithreading techniques to a superscalar processor.

Laudon et al., “Interleaving: A Multithreading Technique Targeting Multiprocessors and Workstations”, has taught a fine-grained multithreaded processor.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Benjamin P. Geib whose telephone number is (571) 272-8628. The examiner can normally be reached on Mon-Fri 8:30am-5:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kim Huynh can be reached on (571) 272-4147. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Benjamin P Geib
Examiner
Art Unit 2181

3/6/06

HENRY W. H. TSAI
PRIMARY EXAMINER